# Automatic evolution of molecular nanotechnology designs

Al Globus, MRJ Technology Solutions, Inc. at NASA Ames Research Center

John Lawton, University of California at Santa Cruz

Todd Wipke, University of California at Santa Cruz

## Design problem

Macroscopic systems built from molecular nanotechnology components are expected to be extremely complex since the components are only a few nanometers across. For example, a product with a volume of one cubic meter would contain $10^{24}$ 10nm components. To design systems of this complexity, highly automated design tools may be of great utility.

There are several classes of molecular nanotechnology designs that can be described as graphs; i.e., a set of vertices and a set of edges each of which connects two vertices. Molecules can be described as a set of atoms (vertices) connected by a set of bonds (edges). Analog circuits can be described as a set of vertices (nodes) connected by a set of wires or components (edges). Digital circuits and, presumably, future nanoelectronic circuits can be similarly described. An automated system for designing graphs with desirable properties should therefore be able, at least in principle, to design a variety of molecular nanotechnology systems.

## Genetic programming

One approach to automated design that has achieved substantial success is genetic programming [Koza 92]. To solve a problem using genetic programming the engineer creates a test that can rate designs (the fitness function), invents or finds a set of hierarchical functions and terminals that can describe designs (the tree nodes), chooses various system parameters such as population size, and lets the genetic programming software run. The software generates a number of individual programs (the population) in the chosen language at random. These individuals are rated by the fitness function. Pairs of programs (the parents) are then selected at random with a bias for high scoring individuals. A subtree is selected from each parent at random and these subtrees are exchanged (crossover) to create two new individuals in the next generation. The software generally runs for a fixed number of generations or until a "good enough" design is found. Note that the procedure is almost embarrassingly parallel making genetic programming an excellent application for parallel computers and networks of workstations or PCs.

## Genetic programming design of analog circuits

Genetic programming has been used to design a variety of analog circuits [Koza 97]. A tree language to generate analog circuits compatible with the SPICE (Simulation Program with Integrated Circuit Emphasis) [Quarles 94] simulator was constructed and a 64 node (80MHz per node) parallel supercomputer was used to design the circuits. The system designed a lowpass filter, a crossover filter, a four-way source identification circuit, a cube root circuit, a time-optimal controller circuit, a 100 dB amplifier, a temperature-sensing circuit, and a voltage reference source circuit. Some of the resulting circuits were comparable in quality to hand designed circuits. NASA Ames has also experimented with evolving linear programs to generate analog circuits with some success [Lohn 98].

## Genetic graphs

Our present work investigates evolving graphs directly (genetic graphs) and comparing the results with genetic programming on the same problem. For genetic graphs, a population of graphs is generated at

random and evolved by choosing two graphs to be parents, splitting both by eliminating an appropriate set of edges (the cut set), swapping subgraphs, then connecting the new graphs at random among the previously split edges. We then compare the results of genetic graph runs (experiment) with genetic programming runs (control) on the same problem. To minimize differences between experiment and control, the initial random population is generated from trees created at random in both cases. Our initial problem is drug design where the fitness function is similarity to a known to drug such as morphine. An atom pair similarity test [Carhart 85] is used in the fitness function. Future planned work includes nanoelectronic circuit design, especially carbon nanotube circuits as simulators and construction algorithms become available.

# References

[Carhart 85] Raymond Carhart, Dennis H. Smith, and R. Venkataraghavan, "Atom pairs as molecular features in structure-activity studies: definition and application," *Journal of Chemical Information and Computer Science*, 23, pages 64-73, 1985.

[Lohn 98] Jason D. Lohn and Silvano P. Colombano, "Automated analog circuit synthesis using a linear representation," submitted ICES-98, 1998.

[Quarles 94] T. Quarles, A. R. Newton, D. O. Pederson, and A. Sangiovanni-Vincentelli, *SPICE 3 Version 3F5 User's Manual*, Department of Electrical Engineering and Computer Science, University of California at Berkeley, CA, March 1994.

[Koza 92] John R. Koza, *Genetic Programming: on the programming of computers by means of natural selection*, MIT Press, Massachusetts, 1992.

[Koza 97] John R. Koza, Forrest H. Bennett III, David Andre, Martin A. Keane and Frank Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Transactions on Evolutionary Computation*, volume 1, number 2, pages 109-128, July 1997.